1.1 OBJETIVO DE LOS SISTEMAS DE BASE DE DATOS

- **DBMS: DATA BASE MANAGEMENT SYSTEM (SISTEMA DE MANEJO DE BASE DE DATOS).**
- **DBM: DATA BASE MANAGER (MANEJADOR DE BASE DE DATOS).**
- **DB: DATA BASE (BASE DE DATOS).**

Un DBMS consiste de una base de datos y un conjunto de aplicaciones (programas) para tener acceso a ellos. Comúnmente, la base de datos contiene información interrelacionada y referente a una misma entidad o empresa.

El objetivo primordial de un DBMS es crear un ambiente en el que sea posible almacenar y recuperar información en forma eficiente y conveniente.

Otro modelo que se utiliza comúnmente para manipular una base de datos es el llamado **SISTEMA DE PROCESAMIENTO DE ARCHIVOS**; que consta de un conjunto de programas que permiten el acceso a la base de datos, pero no optimizan los métodos utilizados, provocando entre otros los siguientes problemas:

- **REDUNDANCIA.** Esta se presenta cuando se repiten innecesariamente datos en los archivos que conforman la base de datos. Esta redundancia aumenta los costes de almacenamiento y acceso y además puede llevar a inconsistencia de los datos.
- **INCONSISTENCIA.** Ocurre cuando existe información contradictoria o incongruente en la base de datos.
- **DIFICULTAD EN EL ACCESO A LOS DATOS.** Debido a que los sistemas de procesamiento de archivos generalmente se conforman en distintos tiempos o épocas y ocasionalmente por distintos programadores, el formato de la información no es uniforme y se requiere de establecer métodos de enlace y conversión para combinar datos contenidos en distintos archivos.
- **AISLAMIENTO DE LOS DATOS.** Se refiere a la dificultad de extender las aplicaciones que permitan controlar a la base de datos, como pueden ser, nuevos reportes, utilerías y demás debido a la diferencia de formatos en los archivos almacenados.
- **ANOMALIAS EN EL ACCESO CONCURRENTE.** Ocurre cuando el sistema es multiusuario y no se establecen los controles adecuados para sincronizar los procesos que afectan a la base de datos. Comúnmente se refiere a la poca o nula efectividad de los procedimientos de bloqueo.
- **PROBLEMAS DE SEGURIDAD.** Se presentan cuando no es posible establecer claves de acceso y resguardo en forma uniforme para todo el sistema, facilitando así el acceso a intrusos.
- **PROBLEMAS DE INTEGRIDAD.** Ocurre cuando no existe a través de todo el sistema procedimientos uniformes de validación para los datos.

1.2 ABSTRACCIÓN DE LA INFORMACIÓN

Uno de los objetivos del DBMS es el ocultar al usuario final ciertos aspectos técnicos relativos al diseño de los manejadores, puesto que estos no son relevantes para este usuario final.

Las estructuras de datos utilizadas para el almacenamiento y recuperación de la información son muchas veces altamente complejas con el objeto de crear un sistema eficiente.

Para diferenciar las etapas en que cada operador, diseñador y otros personajes intervienen, debe distinguirse entre los siguientes niveles de diseño:

1. **NIVEL FISICO.** Es aquel en el que se determinan las características de almacenamiento en el medio secundario. Los diseñadores de este nivel poseen un

amplio dominio de cuestiones técnicas y de manejo de hardware. Muchas veces se opta por mantener el nivel físico proporcionado por el sistema operativo para facilitar y agilizar el desarrollo.

2. **NIVEL CONCEPTUAL**. Es aquel en el que se definen las estructuras *lógicas* de almacenamiento y las relaciones que se darán entre ellas. Ejemplos comunes de este nivel son el diseño de los registros y las ligas que permitirán la conexión entre registros de un mismo archivo, de archivos distintos incluso, de ligas hacia archivos.
3. **NIVEL DE EDICION**. Es aquel en el que se presenta al usuario final y que puede combinaciones o relaciones entre los datos que conforman a la base de datos global. Puede definirse como la forma en el que el usuario aprecia la información y sus relaciones.

1.3 MODELOS DE DATOS

Un modelo de datos es un conjunto de herramientas conceptuales para describir los datos, las relaciones entre ellos, su semántica y sus limitantes.

Los modelos de datos se clasifican en tres grupos principales:

1. **MODELOS LOGICOS BASADOS EN OBJETOS.** Son aquellos que nos permiten una definición clara y concisa de los esquemas conceptual y de visión. Su característica principal es que permiten definir en forma detallada las limitantes de los datos. Ejemplos de este tipo de modelos son:
   - Modelo entidad relación.
   - Modelo binario
   - Modelo semántico de los datos
   - Modelo infológico
2. **MODELOS LOGICOS BASADOS EN REGISTROS.** Operan sobre niveles conceptual y de visión. Sus características principales son que permiten una descripción más amplia de la implantación, pero no son capaces de especificar con claridad las limitantes de los datos. Son ejemplos de este tipo de modelos:
   - **Modelo relacional:** Los datos y las relaciones se representan mediante tablas, cada una con diferentes columnas y nombres únicos.
   - **Modelo de red:** Los datos se representan mediante nombres de registros y las relaciones mediante conjunto de ligas.
   - **Modelo jerárquico:** Es semejante al modelo de red, pero con una estructura arbolada.

3. **MODELOS FISICOS DE DATOS.** Describen los datos en el nivel más bajo y permiten identificar algunos detalles de implantación para el manejo del hardware de almacenamiento. Ejemplos de este tipo de modelos son:

   - Modelo unificador
   - Modelo memoria de cuadros

1.4 INSTANCIAS Y ESQUEMAS

Como es obvio, la base de datos es dinámica y por tanto se encuentra sujeta a modificaciones constantes por la agregación, eliminación y alteración de datos.

Para definir las distintas etapas por las que atraviesa una base de datos, se utiliza el concepto *de* ***instancia de la base de datos***; esta se refiere al estado que ésta guarda en un momento determinado.

También es relevante el observar que uno o varios archivos pudieron ser sujetos a una reestructuración o reorganización. Para solucionar el problema referente al estado que

guarda la estructura de la base de datos, se define el concepto de *esquema de la base de datos;* este hace referencia al estado que guarda la organización conceptual (estructura, ligas, relaciones y demás) de la base de datos en un momento determinado.

REESTRUCTURACIÓN. Cuando se hacen cambios en la estructura, quitar un campo, agregar, modificar longitud o un tipo.
REORGANIZACIÓN. Cambiar el modelo con el que se controla el acceso a los datos.

NOTA: Comúnmente la reorganización trae como consecuencia la reestructuración.

Existen varios esquemas para cada nivel de la base de datos; de tal forma, tenemos un esquema físico, esquema conceptual y esquema de visión (llamado también subesquema).

1.5 INDEPENDENCIA DE LOS DATOS

Esta se refiere a la libertad que pueda existir para modificar algunos de los esquemas sin que exista la necesidad de reescribir los programas de aplicación.

Existen básicamente dos tipos de independencia:

INDEPENDENCIA FÍSICA. Esta se presenta cuando es posible la modificación del esquema físico sin afectar a los esquemas restantes. Las principales razones para llevar a cabo una modificación del esquema físico serán un ajuste en el hardware de almacenamiento o una redistribución de los datos en él.

INDEPENDENCIA LÓGICA. Ocurre cuando se modifica el esquema conceptual sin afectar al resto de los esquemas. Básicamente se modifica el esquema conceptual cuando cambian las características de los datos a almacenar.

Es relativamente más sencillo y probable lograr la independencia física puesto que una modificación del esquema conceptual, (estructuras, ligas y demás) inevitablemente requerirá de modificaciones el código para su manipulación

1.6 LENGUAJE DE DEFINICION DE DATOS

Un esquema de base de datos se especifica por medio de un conjunto de definiciones que se expresan mediante un lenguaje especial llamado lenguaje de definición de datos.

El resultado de la combinación de sentencias de DDL es un conjunto de tablas las cuales se almacenan en un archivo especial llamado diccionario de datos.

El **DDL** (Data Definition Language) es aquel que permite describir un esquema de base de datos. Las definiciones resultantes conformaran al **DICCIONARIO DE DATOS**.

Un **DICCIONARIO DE DATOS e**s un archivo que contiene metadatos que se consulta antes de leer o modificar datos reales en el sistema de base de datos.

1.7 LENGUAJE DE MANIPULACION DE DATOS

El **DML** (Data Manipulation Language) nos sirve para manejar la información contenida en la base de datos. Este manejo consiste básicamente en la inserción, recuperación, eliminación y modificación de la información.

El **DML** aplicado a nivel físico será utilizado para realizar procesos que permitan un acceso más eficiente a la información; en el nivel de visión tendrá como finalidad mostrar al usuario destino los datos en una forma clara y sencilla.

Existen dos tipos de **DML**:

DE PROCEDIMIENTOS. Especifican cuales datos habrán de ser manipulados y el método que se utilizará para ello.

SIN PROCEDIMIENTOS. Solamente especifican los datos a manejar.

Los **DML** de procedimientos son mucho mas eficientes en lo que respecta a sus capacidades de manejo y control de la información, pero su complejidad es mayor.

Se define una *consulta* como una operación que solicita la recuperación de información. La parte del DML que se encarga de procesar esta recuperación se conoce como ***LENGUAJE DE CONSULTA*.**

1.8 MANEJADOR DE BASES DE DATOS

Es una interface entre los datos de bajo nivel y los programas de aplicación y módulos de consulta que se utilizan a nivel de usuario.

Las funciones del manejador de bases de datos son:

La interacción con el manejador de archivos, esto se lleva acabo traduciendo proposiciones con el DML a instrucciones de bajo nivel para la manipulación de los datos.

Implantación de integridad, se encarga de verificar que durante las actualizaciones no se viole ninguna limitante de consistencia.

Mejoramiento del nivel de seguridad, se encarga de restringir el acceso mediante una serie de password u otros medios de identificación y validación.

Respaldo y recuperación, proporciona medios automáticos o semiautomáticos para el respaldo de la información. Permite también la recuperación del sistema en caso de caídas, restablecimiento el estado original de la base de datos hasta antes de la falla.

Control de concurrencia, supervisa los accesos en un ambiente multiusuario, determinando a qué parte del código y de los datos pueden accesar los usuarios en un momento determinado. El objetivo primordial es mantener la consistencia de la base de datos.

1.9 ADMINISTRADOR DE LA BASE DE DATOS

El DBA es quien tiene el control centralizado de la base de datos. Se persigue con esto reducir el número de personas que tengan acceso a los detalles técnicos y de diseño para la operación del DBMS.
Las soluciones principales de un DBA son:

DEFINICIÓN DEL ESQUEMA. Crea el esquema original de la base de datos y genera el diccionario de datos por medio de proposiciones en DDL.
DEFINICIÓN DE ESTRUCTURAS DE ALMACENAMIENTO Y MÉTODOS DE ACCESO. Se encarga de generar o seleccionar estructuras para el medio secundario y definir los métodos de acceso a la información, esto último por medio de proposiciones en DML.

MODIFICACION DE ESQUEMA Y ORGANIZACIÓN. Es una actividad poco frecuente que consiste en rediseñar el esquema de la base de datos. Esto se haría necesario ante la modificación abrupta de las condiciones originales que dieron pie al diseño del esquema primario. Las proposiciones para llevar a cabo esta tarea se realizan en DDL.
CONCESIÓN DE AUTORIZACIONES DE ACCESO. Se encarga de registrar a los usuarios para permitir su acceso al DBMS. Asigna a cada uno de ellos una serie de atributos que le permiten gozar de privilegios como el acceso a determinadas áreas de aplicación, de los datos o del uso de recursos en el sistema.
ESPECIFICACIÓN DE LAS LIMITANTES DE INTEGRIDAD. Crea una serie de tablas donde se especifica el conjunto de restricciones que serán aplicables durante los procesos de actualización

1.10 USUARIOS DE LA BASE DE DATOS

El objetivo primordial de un sistema de base de datos es proporcionar un entorno para recuperar información y almacenar nueva información en la base de datos.

Las personas tienen acceso DBMS se clasifican de la siguiente manera:

USUARIOS INGENUOS. Son aquellos que interactúan con el sistema por medio de aplicaciones permanentes.
USUARIOS SOFISTICADOS. Son aquellos con la capacidad de acceder a la información por medios de lenguajes de consulta.
PROGRAMADORES DE APLICACIÓN. Son aquellos con un amplio dominio del DML capaces de generar nuevos módulos o utilerías capaces de manejar nuevos datos en el sistema.
USUARIOS ESPECIALIZADOS. Son aquellos que desarrollan módulos que no se refieren precisamente al manejo de los datos, si no a aplicaciones avanzadas como sistemas expertos, reconocimientos de imágenes, procesamiento de audio y demás.

1.11 ESTRUCTURA GENERAL DEL SISTEMA

Un sistema de base de datos se divide en módulos que tratan cada una de las responsabilidades del sistema general. En la mayoría de los casos, el sistema operativo del computador proporciona únicamente los servicios más básicos, y el sistema de la base de datos debe partir de esa base. Así el diseño de un sistema de base de datos debe incluir la consideración de interfaz entre el sistema de base de datos y el sistema operativo.

El sistema global puede verse como la agrupación e interacción de los siguientes elementos:

MANEJADOR DE ARCHIVOS. Asigna espacio en el medio de almacenamiento para las estructuras que habrán de almacenar la información.
MANEJADOR DE BASE DATOS. Es la interface entre los datos de bajo nivel y los programas de aplicaciones.
PROCESADOR DE CONSULTAS. Se encarga de traducir las proposiciones de un lenguaje de consultas a instrucciones de bajo nivel.
PRECOMPILADOR DE DML. Se encarga de traducir las proposiciones en DML al lenguaje de diseño del manejador (Pascal, C, Ensamblador etc.).
COMPILADOR DE DDL. Se encarga de convertir las proposiciones en DDL a tablas que contienen metadatos.

Las estructuras de datos requeridas para la operación del DBMS son:

ARCHIVOS DE DATOS. Almacenan a la base de datos.
DICCIONARIO DE DATOS. Almacenan información referente a la estructura de la base de datos.
ÍNDICES. Permiten un acceso eficiente (rápido y confiable) a la información almacenada en la base de datos.

Unidad II:
Modelo Entidad_Relación

2. MODELO ENTIDAD-RELACIÓN

Es uno de los modelos lógicos basados en objetos y por lo tanto se enfoca primordialmente a los niveles conceptual y de visión. Una de las características de este modelo es que permite representar con claridad las limitantes de los datos. El modelo **Entidad-Relación** es en esencia una herramienta para representar el mundo real por medio de simbologías y expresiones determinadas.

2.1 ENTIDADES Y CONJUNTOS DE ENTIDADES

Una entidad es un objeto que existe y puede ser distinguido de otro objeto. Una entidad puede ser concreta (un libro, un automóvil etc.) o abstracta (fecha, edad, etc.).

Un conjunto de entidades es un grupo de entidades del mismo tipo. Una entidad puede pertenecer a mas de un conjunto de entidades a la vez. Por ejemplo, la entidad persona puede ser parte de los conjuntos de entidades alumnos, empleados, clientes etc.

Una entidad se distingue de otra porque posee ciertas características que la hacen única. A estas características se les conoce como atributo. El rango de valores validos para un atributo determinado será conocido como dominio del atributo.

Ejemplo:

Entidad Empleado X

Atributo :

-RFC
-Nombre
-Salario (2000..10,000)
-Edad (18..60)

Una entidad se describe por un conjunto de parejas en el siguiente formato (atributo, valor del dato); debiendo especificarse una pareja por cada atributo de la entidad.

Ejemplo:

{(Nombre,Juan), (Edad,15), (Carrera,LI) }

2.2 RELACIONES Y CONJUNTOS DE RELACIONES

Una relación es una asociación entre varias entidades. Un conjunto de relaciones un grupo de relaciones del mismo tipo.

La mayoría de las relaciones son BINARIAS; no obstante, pueden existir relaciones que incluyan a mas de dos conjuntos de entidades.

Normalmente asocian a dos conjuntos de entidades y la relación tendrá una función determinada; a esta se le denomina papel. Normalmente se utilizan los papeles para etiquetar y así reconocer las relaciones establecidas.



Las relaciones también pueden tener atributos descriptivos, en cuyo caso, la relación se describe indicando la pareja (atributo, ultimo valor del atributo) sobre la relación.

2.3 LIMITANTES DE MAPEO

El modelo **E-R** permite definir una serie de limitantes aplicables en la información contenida en la base de datos básicamente, pueden definirse dos tipos de limitantes:

a) CARDINALIDAD DEL MAPEO.- es aquella mediante la cual puede especificarse la cantidad de entidades que podrán asociarse mediante una relación.

La CARDINALIDAD del mapeo se aplica generalmente sobre dos conjuntos de entidades.

Las cardinalidades existente para dos conjuntos de entidades A y B y conjunto de relaciones R pueden ser:

1. UNA A UNA: Una entidad de A puede asociarse únicamente con una entidad de B.
2. UNA A MUCHAS: Una entidad de a puede asociarse con cualquier cantidad de entidades de B.
3. MUCHAS A UNA: Cualquier cantidad de entidades de A puede asociarse con una entidad de B.
4. MUCHAS A MUCHAS: Cualquier cantidad de entidades de a puede asociarse con cualquier cantidad de **entidades en B.**

Ejemplo:

UNA A UNA	UNA A MUCHAS	MUCHAS A UNA	MUCHAS A MUCHAS
Alumnos Tesis	Carreras Alumnos	Alumnos Carreras	Alumnos Materias
A B	A B	A B	A B



b) DEPENDENCIA DE EXISTENCIA.- Nos permiten definir que un conjunto de entidades esta condicionado a la existencia de otro un ejemplo de este condicionamiento se da entre una entidad alumno y la entidad calificación.

A esta limitante se le denomina dependencia por existencia. Si una entidad Y requiere de una entidad X para existir se dice que Y es dependiente por existencia de X; est implica que si eliminamos a la entidad X; deberá eliminarse la entidad Y.

Para el caso anterior, se nombrara a X como la entidad dominante, y a Y como entidad subordinada

2.4 LLAVES PRIMARIAS

Uno de los procesos de mayor relevancia en la manipulación de una base de datos es el de distinguir entre las diversas entidades y relaciones que son manipuladas. Entendemos como una llave al medio que nos permite identificar en forma unívoca (única e inequívoca) a una entidad dentro de un conjunto de entidades.

Existen diversas categorías que permiten clasificar los tipos de llaves a utilizara:

a) SUPER -LLAVE .- Es un conjunto de atributos mediante los cuales es posible reconocer a una entidad. Este tipo de llaves contiene comúnmente atributos ajenos; es decir; atributos que no son indispensables para llevar a cabo el reconocimiento del registro.

Ejemplo:

Conjunto de entidades:

Cursos

Atributos Super llaves

*Nombre materia Nombre, mat, carrera, semestre

*Carrera Nombre, mat, carrera, unidades

*Semestre Nombre, mat, carrera, semestre, periodo

*Periodo Nombre, mat, carrera

*Unidades

-Si el conjunto de atributos X es una super llave entonces cualquier conjunto de X será super-llave.

b) LLAVE CANDIDATO.- Son aquellas super llaves que no contienen atributos ajenos; es decir, aquellos conjuntos de atributos que no tienen un subconjunto menor que pueda considerarse como super llave.

c) LLAVE PRIMARIA.- Es aquella llave que el diseñador de la base de datos selecciona entra las llaves candidatos encontradas.

Existen conjuntos de entidades que no poseen los atributos necesarios para conformar una llave primaria; se les conoce como entidad débil. Cuando existen los atributos necesarios para formar una llave primaria, se denominan entidad fuerte. Las entidades débiles se subordinan a las entidades fuertes.

Ejemplo:

Fuerte Débil Fuerte



Nota: ncred y clave son las llaves primarias.

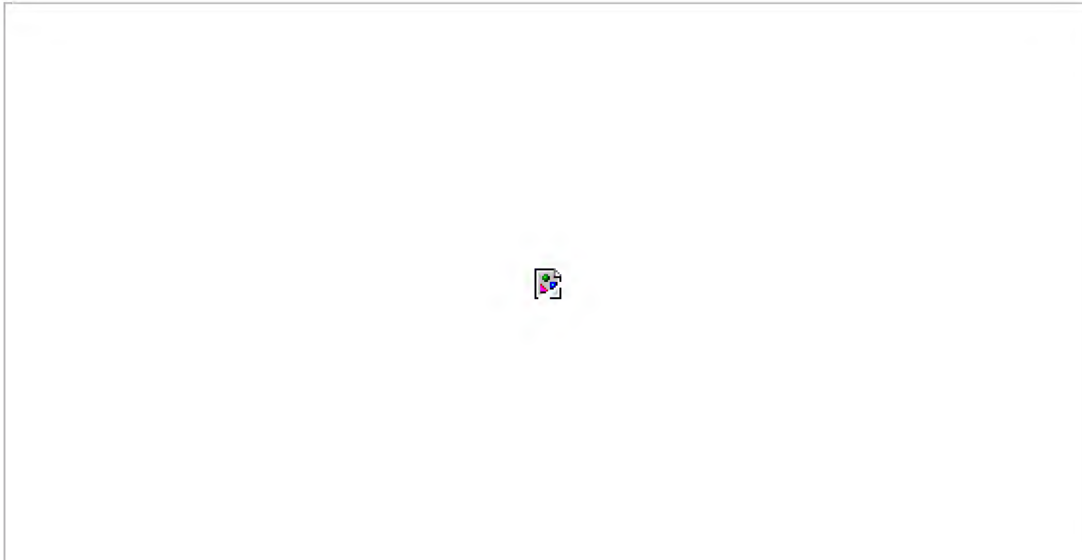
En el ejemplo anterior se pretende que el proceso de renta es una entidad abstracta para clarificar el concepto de entidad débil. No obstante, la mejor implementaron consiste en manipular a la renta como una relación.

Las entidades débiles no pueden ser conocidas por sí solas; con el objeto de diferenciarlas se seleccionan algunos de sus atributos para formar un discriminador. Este discriminador se asocia con las llaves primarias de las entidades fuertes a las que se encuentre subordinada para formar así su llave primaria propia.

Los conjuntos de relaciones también tienen llaves primarias. Estas se conforman por las llaves primarias de los conjuntos de entidades que se asocian en la relación y todos los atributos descriptivos de la relación.

2.5 DIAGRAMAS DE ENTIDAD - RELACIÓN

Son esquemas que nos permitan representar conjunto de entidades y sus relaciones mediante la siguiente simbología.



* Conjunto de entidades o relación con sus atributos
* Conjunto entidades con relaciones
* Cada elemento debe etiquetarse con su nombre.

CARDINALIDAD DE LAS RELACIONES



Notas:

a) Las entidades débiles se señalan como rectángulos de doble pared

b) Los papeles se indican etiquetando las líneas que conectan a los rectángulos con los rombos.

Ejercicios:

Represente mediante Diagramas E-R las siguientes situaciones:

-- Un vídeo club mantiene el control de sus clientes utilizando los siguientes datos: numero de credencial, nombre, dirección y teléfono; él catalogo de películas contiene para cada cassette los datos clave, titulo, clasificación y costo de renta.

A fin de imprimir los pagares y mantener un control de rentas, se registran también las fechas de renta y la cantidad de días que el cliente mantendrá la película.



CONJUNTO DE RELACIONES CON DERIVACIÓN MÚLTIPLE

Puede darse el caso de que una relación sea binaria: es decir, que asocie a mas de dos conjunto de entidades. En estos casos la única variación para representar el modelo consiste en que se establecerá CARDINALIDAD para cada pareja de conjuntos de entidades.



-- En un almacén se lleva el control de los artículos que son vendidos y facturados. El objetivo primordial además de mantener la información almacenada consisten en proceso de facturación. Los datos que se registran: RFC del cliente, nombre del cliente, domicilio, clave del articulo, descripción, costo unitario, numero de factura, fecha, cantidad de artículos vendidos (de cada uno).



2.6 REDUCCIÓN DE DIAGRAMAS E-R A TABLAS.

Con el objeto de observar instancias de las bases de datos, los diagramas E-R se convierten en tablas, Se obtiene una tabla por cada conjunto de entidades o de relaciones.

Existen reglas bien definidas para la conversión de los elementos de un diagrama E-R a tablas:

a) ENTIDADES FUERTES.- Se crea una tabla con una columna para cada atributo del conjunto de entidades.

b) ENTIDADES DÉBILES.- Se crea una tabla que contiene una columna para los atributos que forman la llave primaria de la entidad fuerte a la que se encuentra subordinada.

c) RELACIÓN.- se crea una tabla que contiene una columna para cada atributo descriptivo de la relación y para cada atributo que conforma la llave primaria de las entidades que están relacionadas.

Convierta a tablas y muestre instancias donde pueda observarse la CARDINALIDAD del diagrama E-R en el caso del vídeo club.

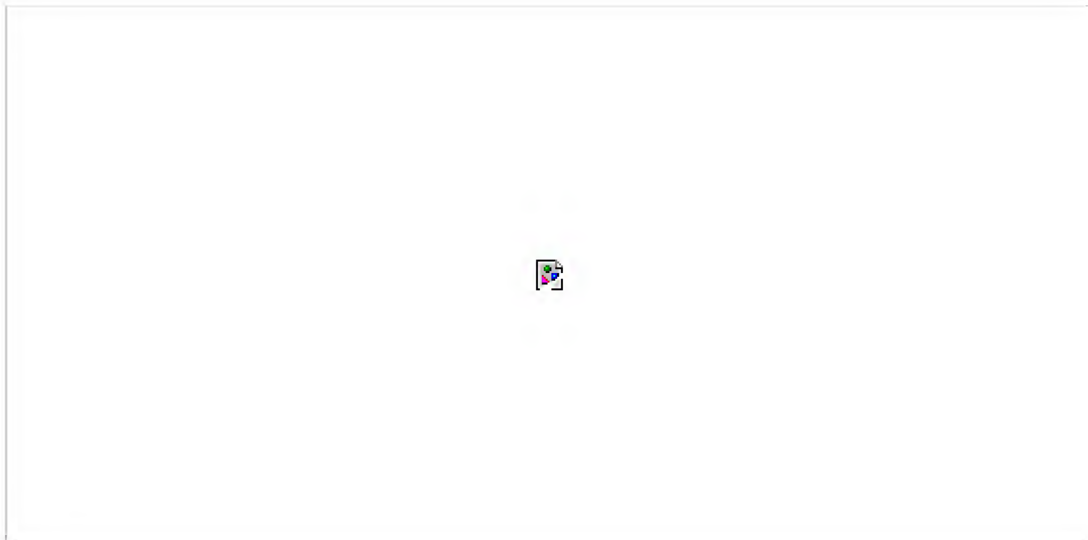




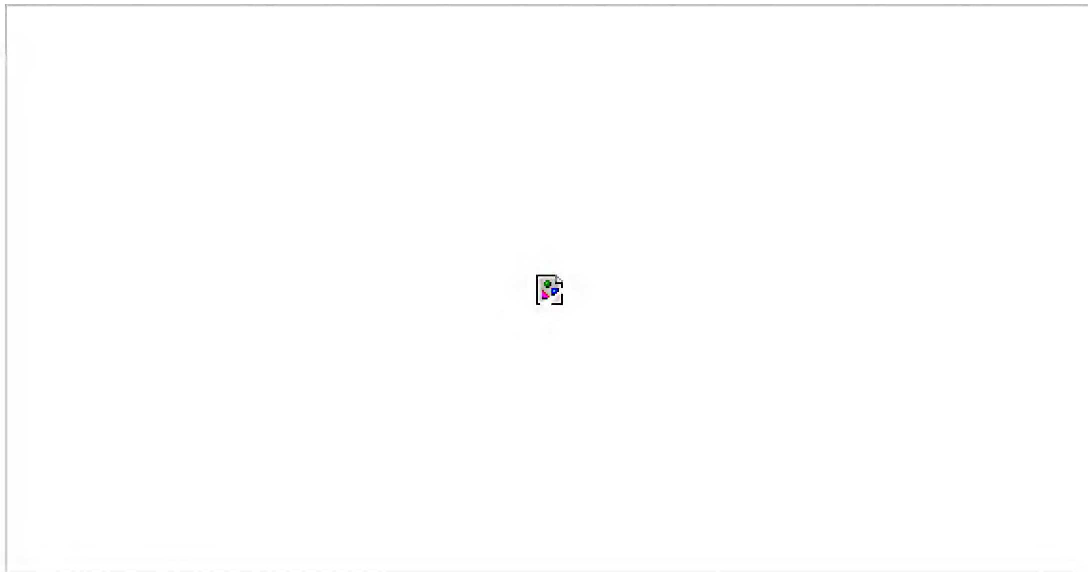


2.7 GENERALIZACIÓN Y ESPECIALIZACIÓN

Son procesos que tienen por objeto la fusión o descomposición de atributos que conforman entidades. La generalización persigue la minimizaron de redundancia en la base de datos de tal manera que puedan ocultarse las diferencias entre entidades formando así entidades comunes.



La especialización en el proceso inverso de la generalización; tiene por objeto reducir el espacio de almacenamiento requerido por la base de datos en el medio físico. Trae como consecuencia una redundancia necesaria, pero suprime el gasto de espacio en el medio secundario para aquellas columnas que no almacenan información por entidades bien determinadas.



INCONVENIENTES DEL MODELO

Entre las limitaciones que presenta este modelo destacan dos:

-No pueden presentarse relaciones entre conjunto de relaciones.
-No pueden visualizarse instancias mediante los diagramas E-R.

2.8 AGREGACIÓN

Es una técnica que permite representar a un bloque de entidades relacionadas como si fueran un solo conjunto de entidades; permitiendo así la relación entre conjunto de relaciones

Unidad III:
Modelo Relacional

- 3.1 Estructura de las Bases de Datos Relacionales.

MODELO RELACIONAL

Este modelo opera en los niveles conceptual y de vision, y tiene la caracteristica de que los resultados de un diseño muestran caracteristicas bien definidas que seran utilis para la implementación del nivel conceptual.

3.1 ESTRUCTURA GENERAL DE LA BD RELACIONALES



3.2 LENGUAJES FORMALES DE CONSULTA

Existe un sistema de DML conocido como *algebra relacional* que permite especificar operaciones de consulta a traves de pasos intermedios de generación de tablas utilizando formatos especiales. En el algebra relacional no son validos los operadores logicos.

Existen dos clasificaciones de procesos en algebra relacional; en cada uno de ellos se toma una o dos tablas como entrada y se obtiene una tabla de salida.

Estas clasificaciones son:

Operaciones tradicionales

- Union(union)
- Intersección(intersect)
- Diferencia(minus)
- Producto cartesiano (times)

Operadores especiales

- Select
- Project
- Join
- Divide

Operadores tradicionales: Estas requieren que las tablas a operar tengan la misma cantidad de atributos y que sus dominios correspondientes sean semejantes o congruentes.

a. UNION.- Constituye una tabla que contiene a todas las tuplas que aparecen en una o ambas tablas
<tabla1>union<tabla2>
b. INTERSECCION.- Produce un atabla que contiene a aquellas tuplas que aparecen en ambas tablas
<tabla1>intersec<tabla2>
c. DIFERENCIA.-Produce un atabla que contiene todas las tuplas de la primera tabla operando que no aparecen en la

segunda <tabla1>minus<tabla2>

d. PRODUCTO CARTESIANO.- Produce un atabla que contiene todas las posibles concatenaciones entre los elementos de las tablas involucradas <tabla1>times<tabla2>

***Operaciones especiales*:**

a. SELECT.- Es un formato qu epermite extraer tuplas que satisfacen una condición <tabla>where<condición>
b. PROJECT.- Es un formato que nos permite filtrar atributos en la tabla resultante, especificando aquellos que se desea obtener. <tabla>[<lista de atributos>]
c. JOIN.- Construye una tabla a partir de dos tablas especificas, obteniendo todas las posibles combinaciones entre los elementos de estas y mostrando aquellas que satisfagan una condición determinada <tabla1>join<tabla2>where<condicion>

Nota: Esta operación es semejante a una consulta sobre tabla global cuando se involucran dos tablas y una condicion de filtro.

d. DIVIDE.-Toma dos tablas una de grado (M+N) y la otra de grado(N). construye una tabla de grado M que contiene todos los valores m de la relacion (M+N) cuyo complemento es igual a todos los valores de la relación de orden N. <tabla1>DIVIDE<tabla2>

Ejercicios:

a. Mostrar una lista que contenga los nombres y costos de las peliculas en existencia:
   videos[nombre,costo]
b. Se desea conocer el nombre y domicilio de todos los clientes infantiles
   (clientes where estado = 'infantil')[nombre,domicilio]
c. Se desea conocer el nombre de las peliculas que no son para niños
   (videos where clasificacion ¬ = 'a')[titulo]
d. Se desea la lista de los titulos que cuestan mas de 10.00 de peliculas para adultos
   ((videos where costo>10.00)where clasif='c')[titulo]

3.3 LENGUAJES COMERCIALES DE CONSULTA.

En forma comercila existen diversos paquetes y/o lenguajes mediante los cuales se puede construir un modelo relacional. El lenguaje que se considera estandar para este tipo de aplicaciones es el SQL(structured query languaje), este lenguaje de consulta estructurado proporciona formatos y sintaxis para la manipulación y definición de los datos.



FORMATOS SQL

DDL

Crear Tablas

CREATE TABLE<nombre de la tabla>
(
<campo1> (<tipo>[,NO NULL]),
<campo2> (<tipo>[,NO NULL]),...
)

Tipos Validos

- CHAR (<LONG>)[VAR]
- FLOAT
- INTEGER
- SMALLINT

Crear Indices

CREATE [UNIQUE] INDEX <nom.indice> ON < nom>tabla>
(
<nomcampo1> [ASC/DES],
<nomcampo2> [ASC/DES],...
)

Modificar Tablas (expandirlas)

- EXPAND TABLE <nom.tabla>
- ADD FIELD <nom.campo> (<TIPO>[NO NULL])

Eliminar Tablas

- DROP TABLE <nom.tabla>

Borrar Indices

- DROP <nom>indice>

Tablas

CREATE TABLE persona
(
nombre(CHAR(40) VAR, NO NULL),
edad(SMALLINT, NONULL),
estatura(FLOAT, NO NULL),

```
telefono(CHAR(7))
)
```

Indices

Por nombre

```
CREATE INDEX ind_nom ON persona
(
nombre
)
```

Por estatura sin llaves repetidas, descendente

```
CREATE UNIQUE INDEX ind_est ON persona
(
estatura desc
)
```

Por edad(primero mas jovenes); edades repetidas, poe estatura(primero la mas alta)

```
CREATE INDEX ind_ed_est ON persona
(
edad,estatura desc
)
```

Modificaciones

- EXPAND TABLE persona
- ADD FIELD direccion(CHAR(30), VAR, NO NULL)

DML

Insertar datos

```
INSERT INTO <nom.tabla>
[(<campo1>,<campo2>...):]
< <valor1>,<valor2>...>
```

Modificar datos

```
UPDATE <nom.tabla>
SET <campo1> = <campo1>,
<campo2> = <campo2>,...
[WHERE <condicion>]
```

Eliminar datos

```
DELETE<nom.tabla>
[WHERE <condicion>]
```

CONSULTAS

En una tabla

```
SELECT [UNIQUE] <lista de campos/*>
FROM <nom>tabla>
[WHERE <condicion>]
[ORDER BY <campo> [asc/des]]
```

Funciones integradas en select/where

```
COUNT(*) conteo
SUM(<campo>) total(acumulador)
AVG(<campo>) promedio
MAX(<campo>) maximo
MIN(<campo>) minimo
```

Ejemplo:

Insert into persona
< 'juan',15,1,75,'2-15-15','forjadores'>

- **Íncrementar edad de todas las tuplas**

```
Update persona
Set edad=edad+1
```

- **Personas menores a 20 años crecieron 10 centimetros**

  Update persona
  Set estatura = estatura+0.10
  Where edad<20



Consulta por coincidencia parcial en cadenas

Select <nombre_tabla>
where<campo char>LIKE <cadena de coincidencia>
caracteres validos en <cadena de coincidencia>

"-" un carácter cualquiera.
"%" una secuencia de caracteres cualquiera.



Create table clientes
(
nc(integer,NO NULL),
nombre(char(20)VAR,NO NULL)
domicilio(char(40)VAR,NO NULL)
estado(char(15)VAR,NO NULL)

```
)
insert into clientes
<320,'juan','forjadores','infantil'>
<145,'pedro','catolica','adulto'>
create tabla videos
(
clave(char(4),NO NULL),
titulo(char(20)VAR,NO NULL),
clasificacion(char(1)VAR,NO NULL),
costo(float,NO NULL)
)
insert into videos
<'A320','la roca','b',12.00>
<'b415','tornado','b',12.00>

Create table renta
(
nc(integer,NO NULL),
clave(char(4),NO NULL),
fecha(char(8),NO NULL),
dias(smallint,NO NULL)
)
Insert into renta
<320,'A716','7/10/97',3>
<320','b716','7/10/97'.3>
```

Ejemplos:

1. Muestre el nombre y estado de los clientes cuyo numero de credencial es mayor a 100

```
select nombre estado
form clientes
where nc>100
```

2. Se desea conocer la cantidad de clientes de cada estado

```
select estado, count(*)
from clientes
grop by estado
```

3. Se desea consultar los nombres de las peliculas que cuestan menos de 15.00 que no son infantiles.

```
Select titulo
From videos
Where (costo < 15.00 and clasificacion ù = "a")
```

Consultas en varias tablas

- **En tabla global**

```
select [unique] <lista campos /*>
from <lista de tablas>
where <condicion>
```

Este tipo de consultas se realiza sobre un atabla global que resulta de todas las combinaciones posibles entre las tuplas

de las tablas involucradas.

La condicion en el formato debe aprovecharse para colocar un filtro que permita que solo tas tuplas o combinaciones de estas que sean requeridas, se muestren; esto se logra por medio de las columnas con valor semejante o de las relaciones establecidas.

Si un campo se encuentra en mas de una tabla, su referencia puede formarse con el formato tabla campo.

Ejemplo:

Se desea conocer la lista de los distintos titulos rentados el 8 oct 97

```
Select unique video.tiulo
From renta,videos
Where (renta.fecha = '8/oct/97' and renta.clave = video.clave)
```

Los discriminadores any/all son opcionales y se pueden combinar con un operador relacional o con in/not in en este tipo de consultas se procesa primeramente la tabla mas interna(tabla2) de la cual obtiene una salida determinada; los datos que se obtienen en esta salida se relacionan mediante el discriminador con los datos de la tabla externa, produciendo asi la salida final.

Ejemplo:

- Se desea mostrar los titulos de las peliculas que han sido rentadas por lo mas de dos dias

```
Select videos.titulo
From videos,renta
Where (videos.clave=renta.clave) and (renta.dias>2)
```

En subconsultas

```
select videos.titulo
form videos
where clave in ( select clave
                 from rentas
                 where renta.dias >2)
```

El formato puede extenderse creando subconsultas en multinivel. Se asume el mismo criterio de resolver a partir de la tabla mas interna e ir relacionando los resultados con la tabla externa inmediata sucesivamente.

Ejemplo:

- se desea conocer el domicilio de los clientes que han rentado peliculas para adolescentes.

```
Select clientes.domicilio
Form clientes
Where nc in (select nc
             From renta
             Where clave in (select clave
                             From videos
                             Where clasificacion='b' ))
```

- **En uniones**

Una union permite consultar los resultados de dos o mas tablas en una sola salida; cuando los resultados de las tablas son semejantes (muestran la misma informacion) se suprimen las salidas redundantes, operando asi como una union de conjuntos.

```
Select <lista de campos/*>            Select <lista campos2/*>
From <tabla1>               Union     From<tabla2>
Where<condicin1>                      Where<condicion2>
```

Ejemplo:

- se desea obtener una lista de clientes y de peliculas se desea incorporar solo alos clientes infantiles y a las peliculas que pueden ser rentadas por estos.

```
Select nombre                         Select titulo
From clientes               Union     From videos
Where estado='infantil'               Where clasificacion = 'a'
```

3.4 MODIFICACIÓN DE LA BASE DE DATOS.

La modificación de la base de datos se expresa usando el operador asignación. Las asignacioes se hacen a relaciones ya existentes en la base de datos.

1. **Eliminación.** Una solicitud de eliminación se expresa de forma muy parecida a una consulta. Sin embargo, en vez de presentar tuplas al usuario, quitamos las tuplas seleccionadas de la base de datos. Sólo podemos eliminar tuplas completas; no podemos eliminar únicamente valores de determinados atributos.
2. **Inserción.** Para insertar datos en una relación, bien especificamos la tupla que sa va a insertar o escribimos una consulta cuyo resultado sea un conjunto de tuplas que se va a insertar.
3. **Actualización.** En ciertas ocasiones podemos querer cambiar un valor en una tupla sin cambiar todos los valores de la tupla. Si hacemos estos cambios usando eliminación e inserción, es posible que no podamos conservar los valores que no queremos cambiar. En lugar de ello, usamos el operador actualización

3.5 VISTAS

Son una especie de tablas virtuales; es decir no existen fisicamente sino que forman mediante la selección y/o filtrado de los componentes de otras tablas, una vista puede ser definida en base a una lista previa. Esto significa que pueden crearse dependencia entre las vistas.

Formato de definicion de vistas

DEFINE VIEW<nombre vista>
[(identif_campo1, identif_campo2,...)]
AS<operación de consulta>

· **Ejemplo:** se desea crear una vista para obtener los nombres y domicilios de los clientes adultos es deseable el establecimiento de las cabeceras nombre del cliente, domicilio del cliente.

```
DEFINE VIEW cliente_adulto
(nombre del cliente, domicilio del cliente)
As(select nombre,domicilio
From clientes
Where estado = 'adulto')
```

Como puede verse, la especificación de los identificadores es opcional; si estos se omiten se asumiran los nombres de los campos extraidos en la consulta.

La operación de consulta permite todos los formatos validos de consulta en SQL con execepción del groop by.

Cuando una vista es definida en base a otra, se se dice que es dependiente de esta por lo tanto, se suprimira automaticamente la vista dependiente si se suprime la vista original.

Eliminación de vistas

Drop view <nombre tabla>

- Ejemplo: suponga que se desea crear una vista dependiente de la vista cliente adulto que contenga solamente a los clientes que viven sobre forjadores. Se desean los mismos campos y la vista sera llamada cliente_adulto_forjadores.

```
Define view cliente_adulto_forjadores
AS (select *
From cliente_adulto
Where domicilio_del_cliente like 'forjadores%')
~
~
drop view cliente_adulto
```

(se eliminara tambien la vista cliente_adulto_forjadores, puesto que es dependiente de cliente_adulto.)

La eliminación de una tabla provoca tambien la eliminación automatica de todas las listas que se hayan definido haciendo referencia a ella.

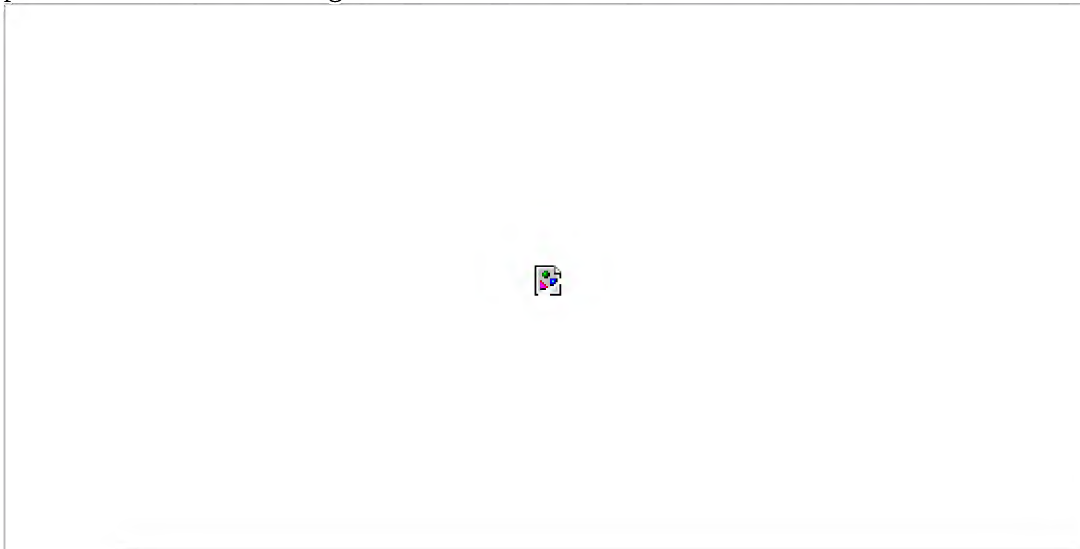
Unidad IV:
Diseño de Bases de Datos Relacionales

4 BASES DE DATOS RELACIONALES

No obstante que pueden desarrollarse sistemas de procesamiento de archivo e incluso manejadores de bases de datos basándose en la experiencia del equipo de desarrollo de software logrando resultados altamente aceptables, siempre es recomendable la recomendación de determinados estándares de diseño que garantizan el nivel de eficiencia mas alto en lo que se refiere a almacenamiento y recuperación de la información. De igual manera se obtiene modelos que optimizan el aprovechamiento secundario y la sencillez y flexibilidad en las consultas que pueden proporcionarse al usuario.

OBJETIVOS DEL DISEÑO DE BASES DE DATOS

Entre las metas más importantes que se persiguen al diseñar un modelo de bases de datos, se encuentran las siguientes que pueden observarse en esta figura.



4.1 PELIGROS EN EL DISEÑO DE BASES DE DATOS RELACIONALES.

Los principales inconvenientes que se presentan cuando el diseño de un modelo no satisface las formas normales son:

* Repetición de la información
* Dificultad para representar y/o interpretar cierta información.
* Perdida de la información

Ejemplo: Suponga que se desea controlar el préstamo de libros a alumnos del tecnológico. Se asume que existe una base de datos de la bibliografía existente cuya llave es la clasificación.



PROBLEMAS

- o Repetición de datos del alumno en cada préstamo
- o No es posible establecer comparativos con usuarios y no usuarios por solo están restringidos los usuarios.
- o Al modificar un atributo del alumno deberá recorrerse todo el archivo...

Versión 2 (ESPECIALIZACIÓN)



- No funciona porque no hay relación.

Versión 3



PROBLEMAS

- No es posible determinar fechas de prestamos de un libro a aun alumno en particular.
- Solo se registran alumnos con prestamos o se desperdicia atributo clasif.
- Repetir datos del alumno por el préstamo o reutilizar el campo clasif, lo ultimo tiene dos problemas:
  - Se pierde historial de prestamos
  - Un alumno solo puede tener un préstamo a la vez.

Al caso mostrado en la Versión 3 se le denomina DESCOMPOSICIÓN CON PERDIDA dado que al realizar la especialización se pierde información que existía por la relación entre los atributos.

Versión 4



- o **MODELO OPTIMO.**

4.2 DEPENDENCIAS FUNCIONALES

Las dependencias funcionales son una restricción al conjunto de relaciones legales. Nos permiten expresar hechos acerca de la empresa que estamos modelando con la base de datos.

Superclave se puede definir como sigue, sea R un esquema de relaciones. Un subconjunto K de R es una superclave de R sí, en cualquier relación legal r(R), para todos los pares t_1 y t_2 de tuplas de r tales que t_1 ¹ t_2, t_1[K]¹ t_2[K]. Es decir, dos tuplas en cualquier relación legal r(R) no pueden tener el mismo valor en el conjunto de atributos K.

La noción de dependencia funcional generaliza la definición de superclave. Sea µÍ R y bÍ R. La dependencia funcional

µ®b se cumple en R si en cualquier relación legal r(R), para todos los pares de tuplas t_1 yt_2 en r tales que t_1[µ]=t_2[µ], también se cumple que t_1[b]=t_2[b].

Utilizando la notación de la dependencia funcional, decimos que K es una superclave de R si K®R. Es decir, K es una superclase sí siempre que t_1[K]=t_2[K]. , también se cumpla que t_1[R]=t_2[R] (es decir, $t_1 = t_2$).

Las dependencias funcionales nos permite expresar restricciones que no pueden expresarse por medio de superclaves. Considérese el esquema siguiente:

Esquema - préstamo = nombre - sucursal, numero - préstamo, nombre - cliente, cantidad.

Ejemplo: si un préstamo se hace a mas de un cliente en este caso a marido/mujer, entonces no esperaríamos que el atributo numero - préstamo fuera una superclave.

APLICACIONES

Las dependencias funcionales se usan de dos formas:

1. - Para especificar restricciones en el conjunto deceleraciones legales. O sea solo se interesa por las relaciones que satisfagan un conjunto dado de dependencias funcionales. Si queremos limitarnos a las relaciones de esquema R que satisfacen F, decimos que F se cumple en R.

2. - Para probar si una relación es legal bajo un conjunto dado de dependencias funcionales. Si una relación r es legal bajo un conjunto F de dependencias funcionales, decimos que r satisface a F.

Considérese la relación r de la siguiente figura y veamos que dependencias funcionales se satisfacen. Obsérvese que A®C se satisface. Hay dos tuplas que tienen el valor de a_1 en A. Estas tuplas tienen el mismo valor en C, c_1´. De manera similar, las dos tuplas con valor a_2 en A tienen el mismo valor en C, c_2´. No existen otros pares de tuplas distintos que tengan el mismo valor en A. Sin embargo, no se satisface la dependencia funcional C®A. Para ver esto considérense las tuplas t_1 = (a_2 ,b_3 ,c_2 , d_3) y t_2 = (a_3 ,b_3 ,c_2 , d_4). Estas dos tuplas tienen el mismo valor en C, c_2´ y distintos valores en A, a_2 y a_3´ respectivamente. Así hemos encontrado un par de tuplas t_1 y t_2 tales que t_1[C]=t_2[C] pero t_1[A]¹ t_2[A]



CIERRE DE UN CONJUNTO DE DEPENDENCIAS FUNCIONALES

No es suficiente considerar un conjunto dado de dependencias funcionales. Además necesitamos considerar todas las dependencias funcionales que se cumplen. Veremos que, dado un conjunto F de dependencias funcionales, podemos probar que se cumplen otras ciertas dependencias funcionales. Se dice que F implica lógicamente dichas dependencias funcionales.

Supóngase que nos dan un esquema de relaciones R = (A, B, C, G, H, I) y el conjunto de dependencias funcionales

A®B

A®C
CG®H
CG®I
B®H

La dependencia funcional

A®H

Se implica lógicamente. Es decir, podemos demostrar que siempre que se cumpla el conjunto dado de dependencias, A®H también debe cumplirse.

TÉCNICAS PARA DEDUCIR DEPENDENCIAS FUNCIONALES

La primera técnica se basa en tres axiomas o reglas de inferencia para dependencias funcionales. Aplicando estas reglas repetidamente, podemos encontrar F^+ completo dado F. En las reglas siguientes, adoptamos el convenio de usar letras griegas (µ ,b ,g ...) para conjuntos adoptamos el convenio de usar letras romanas mayúsculas desde el principio del alfabeto para atributos individuales. Usamos µb para representar µÈb .

REGLAS DE REFLEXIVIDAD. Si µ es un conjunto de atributos y bÍµ entonces se cumple µ®b .

REGLA DE AUMENTO. Si se cumple µ®b y g es un conjunto de atributos, entonces se cumple gµ®gb .

REGLA DE TRANSITIVIDAD. Si se cumple µ®b , y se cumple b®g , entonces se cumple µ®g .

Estas reglas son seguras porque no generan dependencias funcionales incorrectas. Las reglas son completas porque para un conjunto dado F de dependencias funcionales, nos permiten generar F^+ completo. Esta colección de reglas se llama axiomas de A.

Para simplificar mas esta tarea, se listan a continuación algunas reglas adicionales. Es posible utilizar los axiomas de Armstrong para probar que estas reglas son correctas.

REGLA DE UNIÓN .Si se cumplen µ®b y µ®g entonces se cumple µ®bg .

REGLA DE DESCOMPOSICIÓN. Si se cumple µ®bg , entonces se cumplen µ®b y µ®g .

REGLA DE PSEUDOTRANSITIVIDAD. Si se cumplen µ®b y gb®d , entonces se cumple µg®d .

FORMA NORMAL

Se dice que una forma normal particular si satisface cierto conjunto especifico de restricciones; por ejemplo, se dice que una relación esta en primera forma normal si y solo si satisface la restricción de contener únicamente valores atómicos .

A continuación se mencionan las formas normales que existen.

FORMA NORMAL BOYCE-CODD

Una de las formas normales más deseables que podemos obtener es la forma normal Boyce-codd (BCNF). Un esquema de relaciones R esta BCNF con respecto a un conjunto F de dependencias funcionales si para todas las dependencias funcionales en F^+ de la forma µ®b , donde µÍ R y bÍ R, por lo menos se cumple de las siguientes condiciones:

µ®b es una dependencia funcional trivial (es decir, bÍµ).

µ es una superclave del esquema R.

Un diseño de base de datos esta en BCNF si cada una de los miembros del conjunto de los esquemas de relación que comprende el diseño esta en BCNF.

PRIMERA FORMA NORMAL

Una relación R esta en primera forma normal (1NF) y si y solo si todos los dominios subyacentes solo contienen valores atómicos.

Un dominio es atómico si los elementos del dominio se consideran unidades invisibles.

Esta definición trata de decirnos que cualquier relación normalizada esta en 1NF. Una relación que tan solo esta en primera forma normal es decir, una relación en 1NF que, además no esta 2NF y, por tanto, tampoco no esta en 3NF se dice que tiene una estructura indeseable.

SEGUNDA FORMA NORMAL

Una relación R esta en segunda forma normal (2NF) si y solo si esta en 1NF y cada atributo no es primo completamente dependiente de la primera llave primaria.

Un atributo es no primo si no participa en la llave primaria.

TERCERA FORMA NORMAL

En aquellos casos en los que no pueden satisfacerse los tres criterios de diseño, abandonamos BCNF y aceptamos una forma normal más débil llamada TERCERA FORMA NORMAL (3NF). Veremos que siempre es posible encontrar una descomposición de producto sin perdida que conserve las dependencias que estén en 3NF.

BCNF requiere que todas las dependencias no triviales sean de la forma µ®b , donde µ es una superclave.

3NF hace un poco menos estricta esta restricción permitiendo las dependencias funcionales no triviales cuyo lado izquierdo no sea una superclave.

Un esquema de relaciones R esta en 3NF con respecto a un conjunto F de dependencias funcionales si para todas las dependencias funcionales en F^+ de la forma µ®b , donde µÍ R bÍ R, por lo menos se cumple una de las condiciones siguientes.

µb es una dependencia funcional trivial.

µ es una superclave R

Cada atributo A en b¾µ esta contenido en una clave candidata de R.

La definición 3NF permite ciertas dependencias funcionales que nos permiten en BCNF. Una dependencia µ®b que satisface solo la tercera condición de la definición de 3NF no se permite en BCNF, aunque si se permite en 3NF.

CUARTA FORMA NORMAL

Un esquema de relaciones R esta en 4NF con respecto a un conjunto D de dependencias funcionales si para todas las dependencias multivaluadas de D^+ de la forma µ® ®b , donde µÍ R y bÍR, se cumple por lo menos una de las siguientes condiciones:

*µ® ®b es una dependencia multivaluada trivial.

*µ es una superclave del esquema R.

Un diseño de bases de datos esta en 4NF si cada miembro del conjunto de esquema de relaciones que comprende el diseño esta en 4NF.

La analogía entre 4NF y BCNF se aplica al algoritmo para descomponer un esquema en 4NF. La siguiente figura muestra un algoritmo de descomposición en 4NF.

resultado := {R};
listo:=falso;
calcular F^+;
while (not listo) do
if (existe un esquema R_i en resultado que no esta en 4NF)
then begin
sea µ® ®b una dependencia multivaluada no trivial que se cumple en R_i tal que µ® R_i no esta F^+, y
µÇb=Æ ;
resultado:=(resultado - R_i) È(R_i -b)È(µ,b) ;
end;
else listo:=verdadero;

Hemos visto que si nos dan un conjunto de dependencias funcionales y multivaluadas, resulta provechoso encontrar un diseño de bases de datos que se ajuste a los tres criterios siguientes:

* 4NF
* conservación de las dependencias
* Producto sin perdida.

Si todo lo que tenemos son dependencias funcionales, el primer criterio es justo el de BCNF.

Es posible que no se logran los tres criterios antes mencionados, y es aun donde se abandona 4NF, y aceptamos BCNF o incluso 3NF, si es necesario para asegurar la conservación de las dependencias

4.3 FORMA NORMAL PROYECTO -PRODUCTO.

La forma normal de proyecto - producto se define de manera similar a BCNF y 4NF, excepto que se usan dependencias de intersección. Un esquema de relaciones R esta en forma normal de proyecto-producto (PJNF) con respecto aun conjunto D de dependencias funcionales multivaluadas y de intersección si para todas las dependencias en D^+ de la forma*(R_1 , R_2 ...R_n) donde cada R_i Í R y R_1 È R_2È...È R_n´ se cumple por lo menos una de las siguientes condiciones:

*(R_1 , R_2 ...R_n) es una dependencia de producto trivial.
Cada R_i es una superclave de R.

Un diseño de base de datos esta en PJNF si cada miembro del conjunto de esquema de relaciones que comprende el diseño esta en PJNF. PJNF se llama quinta forma normal (5NF).

4.4 FORMA NORMAL DE DOMINIO-CLAVE

La forma norma de dominio-clave esta basada en tres nociones:

DECLARACIÓN DE DOMINIO. Sea A un atributo, sea down un conjunto de valores. La declaración de dominio AÍ down requiere que el valor de A de todas las tuplas sean valores en down.DECLARACIÓN DE CLAVE. Sea R un esquema de relaciones en que KÍ R. La declaración de clave key (K) sea una superclave del esquema R es decir K ® R. Obsérvese que todas las declaraciones funcionales pero no todas las dependencias funcionales son declaraciones de clave.
RESTRICCIÓN GENERAL. Una restricción general es un predicado en el conjunto de todas las relaciones de un esquema dado.

Ejercicios :

Diseñe los modelos que considere mas eficientes para cada uno de los siguientes casos:

a) una empresa desea automatizar su proceso de facturación las facturas contienen un numero de folio, la fecha, los datos del cliente (RFC, nombre, dirección) y de los productos que la factura ampara (clave, descripción, cantidad costo unitario e importe.)

Versión 1

PROBLEMAS

* Se repetirían nombre, RFC, dom.
* En una misma factura se repetirían (fecha, numero de factura).
* No hace falta guardar el importe.

Versión 2

PROBLEMAS
* Repetir datos del cliente por cada factura
* Repite campo RFC y NUMFOL

* No es necesario guardar el importe

Versión 3



PROBLEMAS

* Se repiten datos innecesariamente para cada articulo FECHA y RFC.

Versión 4



PROBLEMAS

* Se graba importe que es innecesario
* Se repetirían para cada factura.

Versión 5



PROBLEMAS

* Se repite fecha y RFC

Versión 6



PROBLEMAS

* No se sabría en que factura va la venta.

Versión 7



PROBLEMAS

* Se repiten CLAVE, DESCRIP y PUNIT para cada venta de ese articulo.

MODELO OPTIMO.

Versión 8



*NOTACIÓN

X® Y (Y depende de X)

b)Dado una relación R, el atributo Y de R es funcionalmente dependiente del atributo X de R si y solo siempre que dos tuplas de R coincidan en sus valores de X también coincidan en sus valores de Y.

Ejemplo:



APLICACIÓN DE LAS Dfs

Existen básicamente tres aplicaciones para las dependencias funcionales:

1.- Determinar si una relación es legal bajo un conjunto dado de dependencias funcionales. Si una relación R es legal bajo un conjunto F de dependencias funcionales, se dice que R satisface a F.

R= Equipos

F_1={1,2,3,4,5,6} R no satisface a F_1

F_2={2,4} R si satisface a F_2

2.- Especificar las limitaciones del conjunto de relaciones legales; de esta manera, se manejaran solamente las relaciones que satisfagan a un conjunto de dependencias funcionales.

3 .- Generación de descomposiciones sin perdida mediante la aplicación de diversos postulados preestablecidos para este fin.

Ejemplo:

X® (Y,Z) Descomposición

X ® Y sin perdida con perdida

X® Z (XYZ) (XYZ)

ßß

(XY) (XZ) (XY) (YZ)

Se dice que una dependencia funcional es trivial cuando todas las relaciones la satisfacen. En general, una dependencia funcional de la forma X® Y es trivial si X es un subconjunto impropio de Y (X y Y se forman de los mismos atributos)

Ejemplo: de dependencias funcionales triviales.

NC® NC

(NC,NOM) ® (NC,NOM)

Una dependencia funcional de la forma X® Y es completa si Y depende funcionalmente de X y no depende funcionalmente de ningún subconjunto propio de X.

(X,Y)® Y

X® Z

X® Y

Ejemplo:

(NC,NOM) ® CARR ES INCOMPLETA

NC® CARR

NC® NOM ES COMPLETA

(NC,CLAVEMATERIA,PERIODO) ® CAL ES COMPLETA

TEORÍA DE DEPENDENCIAS FUNCIONALES

Si se tiene un conjunto de dependencias funcionales, estas pueden implicar que existan otras que también se cumplan.

Sea F un conjunto de dependencias funcionales F^+ es el conjunto cerrado de F que contiene a todas las dependencias funcionales que F implica lógicamente. Dado F podemos obtener F^+; el grado de complejidad para obtener F^+ varia en función del tamaño de F.

Ejemplo:

F={ NC® NOM, NC® CARR, CARR® ESP,...}

F^+{ NC® ESP,...}

TÉCNICAS PARA DEDUCIR DEPENDENCIAS FUNCIONALES

AXIOMAS DE ARMSTRONG

1.- REGLA DE REFLEXIBIDAD. si X es un conjunto de atributos y Y es un subconjunto impropio de X (Y Í X), entonces se cumple que X® Y.

2.- REGLA DE AMPLIFICACIÓN.- si se cumple que X® y entonces debe cumplirse que WX® WY si W es un conjunto de atributos valido en la relación.

3.- REGLA DE TRANSITIVIDAD. si se cumple que X® Y y Y® Z, entonces se cumple que X® Z.

Se dice que estas reglas son validas porque no se generan dependencias funcionales incorrectas. Las reglas son completas porque no dado un conjunto F de dependencias funcionales permiten obtener la totalidad de F^+. Ocasionalmente su aplicación puede darse en forma directa para calculas F^+; en estos casos se ocurre al uso de las siguientes reglas:

1.-REGLA DE UNIÓN. si se cumple X® Y y X® Z, entonces se cumple que X® YZ.

2.- REGLA DE DESCOMPOSICIÓN. si se cumple que X® YZ, entonces se cumple que X® Y y X® Z.

3.- REGLA DE PSEUDOTRANSITIVIDAD. si se cumple X® Y y WY® Z, entonces se cumple WX® Z.

FORMAS NORMALES

La razón de ser de las formas normales consiste en la estandarización de los conceptos relacionados al diseño eficiente de las estructuras y esquemas de una base de datos.

Durante mucho tiempo se ha dependido en extremo de la experiencia y capacidad de los analistas y diseñadores de bases de datos. Como es obvio, existirán discrepancias entre los métodos que estos aplican para obtener un modelo eficiente. Las formas normales permitirán la aplicación de un estándar de eficiencia en niveles ascendentes mediante la aplicación de las mencionadas formas normales.

Se dice que una relación (tabla) esta en una forma normal determinada si satisface cierto conjunto especifico de restricciones.

UNIVERSO DE RELACIONES.



Para avanzar de una forma normal a otra deben verificarse las restricciones de la actual y la nueva forma normal. Una de las herramientas mas utilizadas para alcanzar una nueva forma normal es la DESCOMPOSICIÓN esta debe presentar las siguientes características:

* Debe realizarse sin perdida
* Deben mantenerse las dependencias funcionales
* Se debe evitar o reducir hasta donde sea posible la redundancia.

CASO PRACTICO PARA NORMALIZAR.

Un conjunto de proveedores distribuyen artículos en ciudades especificas. Las ciudades se encuentran agrupadas por regiones, pero un proveedor solo puede cubrir una ciudad. Cada proveedor es capaz de distribuir artículos que están numerados consecutivamente.

La siguiente tabla muestra la distribución de los artículos que son distribuidos por cada proveedor y las existencias actuales de estos.



PRIMERA FORMA NORMAL

Una relación esta en primera forma normal si y solo si los dominios de sus atributos solo contienen valores atómicos (no vas a dejar ningún casillero cío).





SEGUNDA FORMA NORMAL

Una relación esta en segunda forma normal si y solo si esta en primera forma normal y cada atributo no primo* es completamente dependiente de la llave primaria.

*ATRIBUTO PRIMO: es aquel que forma parte de la llave primaria.



}





TERCERA FORMA NORMAL

Una relación esta en tercera forma normal si y solo si esta en segunda forma normal y todo atributo no primo es dependiente no transitivamente de la llave primaria.

No se cumple la tercera forma normal porque hay transitividad de un proveedor a región.



Llave (PROV, ART)





Llave(PROV)





Se considera que un esquema que alcanza tercera forma normal es eficiente. No obstante, se ha propuesto una mejora que permitirá obtener un modelo más eficiente con la ventaja de que no depende de formas normales anteriores (aunque se recomienda ampliamente alcanzar tercera forma normal antes de su aplicación).

BOYCE- CODD (FNBC)

Una relación esta en FNBC si y solo si todas las dependencias funcionales son completas y todos los atributos determinantes son llaves candidatos.

Ejemplo:

Se desea el registro de alumnos; materias y profesores que la imparte. supóngase que un profesor imparte solamente una materia.







NOTAS:

* Una relación que esta en FNBC estará siempre en 3FN; esta relación no es reciproca.
* La forma normal BOYCE-CODD (FNBC) optimiza casos en los que existen varias llaves candidato traslapadas. (son aquellas que comparten atributos.).
* Teóricamente, es más simple puesto que no requiere de formas normales previas.

OBJETIVOS DEL DISEÑO

Se pueden plantear básicamente tres metas al realizar el diseño de un esquema de base de datos;

a)FNBC
b)DESCOMPOSICIÓN DE PRODUCTO SIN PERDIDA
c)CONSERVACIÓN DE LAS DEPENDENCIAS FUNCIONALES.

NOTA:

CASOS ESPECIALES

Puede ocurrir que en algunas situaciones se pierda n dependencias funcionales al alcanzar FNBC; esto trae como consecuencia una reducción en el rendimiento del esquema y pone en riesgo incluso la integridad de la base de datos. En estos casos será preferible conservar el diseño alcanzado hasta 3FN.

DEPENDENCIAS DE VALORES MÚLTIPLES

Existen casos de relaciones en los que un atributo puede determinar a otro restringiendo su rango de valores validos. A este tipo de dependencias se les conoce como dependencias multivaluadas (DMV).

FORMATO:

X èè Y

"X multidetermina a Y"

MATERIA èè PROFESOR



Las formas normales de nivel mas alto (4fn y 5fn) son aplicables para aquellos casos en los que existan dependencias multivaluadas y/o se presenta ciertas características especiales de operación.

CUARTA FORMA NORMAL

Siempre que en una relación R exista una dependencia multivaluada (DMV), la relación esta en 4fn si todos los atributos de R son funcionalmente dependientes del multideterminador

Si:

R = {A,B,C,D}

A B C D

B ® ® D

R esta en 4FN Si

B ® A

B ® C

NOTAS:

* Si una relación esta en 4FN, esta también en FNBC, la relación no es reciproca
* Cualquier relación puede descomponerse sin perdida en un conjunto de relaciones en 4FN.

QUINTA FORMA NORMAL

Una relación esta en 5FN si y solo si toda *dependencia de reunión en R esta implicando por las llaves candidatos de R.

Si en:

R = {A,B,C,D,E}

* Son llaves candidato A y B

*Son dependencias de reunión

{A,B,C}

{A,C,D}

{B,E}

R esta en 5FN

NOTA:

* La quinta forma norma es aplicable comúnmente a aquellos casos en los que realizan descomposiciones hacia 3 o mas nuevas tablas.

Unidad V:

Modelo de Datos de Red

5.1 CONCEPTOS BASICOS

Una base de datos de red se compone por una colección de registras que se conectan entre si por medio de ligas.

Un registro equivale a una entidad y un campo a un atributo del modelo entidad relación. Los campos contienen exclusivamente valores atómicos. Una liga es una relación que se establece solamente entre dos registros; es decir; debe utilizarse una liga para

cada relación entre una pareja de registros.

Ejemplo:

CLIENTES CUENTAS

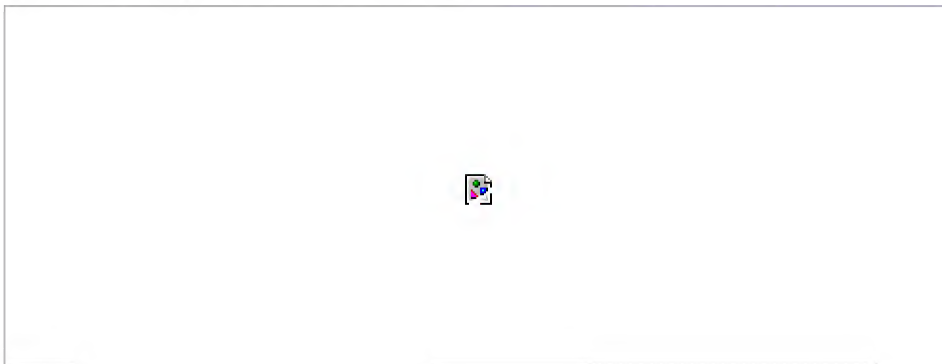


5.2 DIAGRAMAS DE ESTRUCTURAS DE DATOS

Permiten mostrar gráficamente el esquema de una base de datos en el modelo de red. Sus componentes principales son :

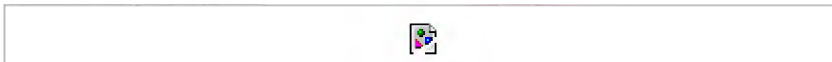
- Cajas o cuadros: representan registros.
- Líneas: representan ligas.

MODELO ENTIDAD - RELACIÓN



RED

CLIENTE CUENTA



El caso anterior muestra la conversión del modelo entidad relación al modelo de red para una relación simple donde no existen atributos descriptivos en la relación.

CASOS ESPECIALES

a) LA RELACIÓN TIENE ATRIBUTOS DESCRIPTIVOS

E-R



D. E. D. RED



b) La relación conecta a mas de dos conjuntos de entidades.
Ejemplo:



D. E. D. RED



5.3 MODELO CODASYL DBTG

Este modelo es ya una implementaron de las reglas generales de operación del modelo de res. Toma de este los aspectos generales operativos, pero introduce las siguientes características particulares:

Solo pueden utilizarse ligas muchos a uno. Se prohiben las ligas muchos a muchos para simplificar la implementaron. Las ligas uno a uno se representan utilizando ligas muchos a uno.

Casos:







CONJUNTOS DBTG

Son representaciones mas concisas del modelo de red; representan genéricamente entre registros.



Una de las aplicaciones mas extendidas de los conjuntos DBTG es la de los *grupos repetidos*. Estos permiten simplificar la representación de atributos que pueden presentar valores múltiples en una instancia determinada.

Ejemplo:

Suponga que un cuenta habiente tiene mas de un domicilio para el envío de sus estados de cuenta y otras notificaciones.



Unidad VI:
Modelo de Datos Jerárquico

6.1 CONCEPTOS BASICOS

Una *base de datos jerárquica consiste en una colección de registros que se conectan entre si por medio de ligas.* Los registros y las ligas son similares a los del modelo de red, pero en el modelo jerárquico se organiza en forma de árbol con raíz (donde la raíz es nodo ficticio); de tal manera que una base de datos jerárquica es una colección de arboles de este tipo, formando un bosque.

A cada árbol con raíz con raíz se le denomina árbol de base de datos.

En este modelo un registro puede tener que repetirse en varios sitios que puede ocasionar los siguientes problemas:

* Riesgos de la inconsistencia al llevar a cabo actualizaciones.
* Inevitable desperdicio de espacio en el medio de almacenamiento secundario.

6.2 DIAGRAMAS DE ESTRUCTURAS DE ÁRBOL

Un diagrama de estructura de árbol es el esquema de una base de datos jerárquica. Tiene dos componentes básicos:

REGISTROS y LIGAS

Estos diagramas son similares a los de estructura de datos en el modelo de red. La diferencia radica en que en el modelo de red los registros se organizan en forma de un grafo arbitrario mientras que en el modelo jerárquico se organiza en forma de un árbol con raíz.

Las reglas para la formación del árbol son:

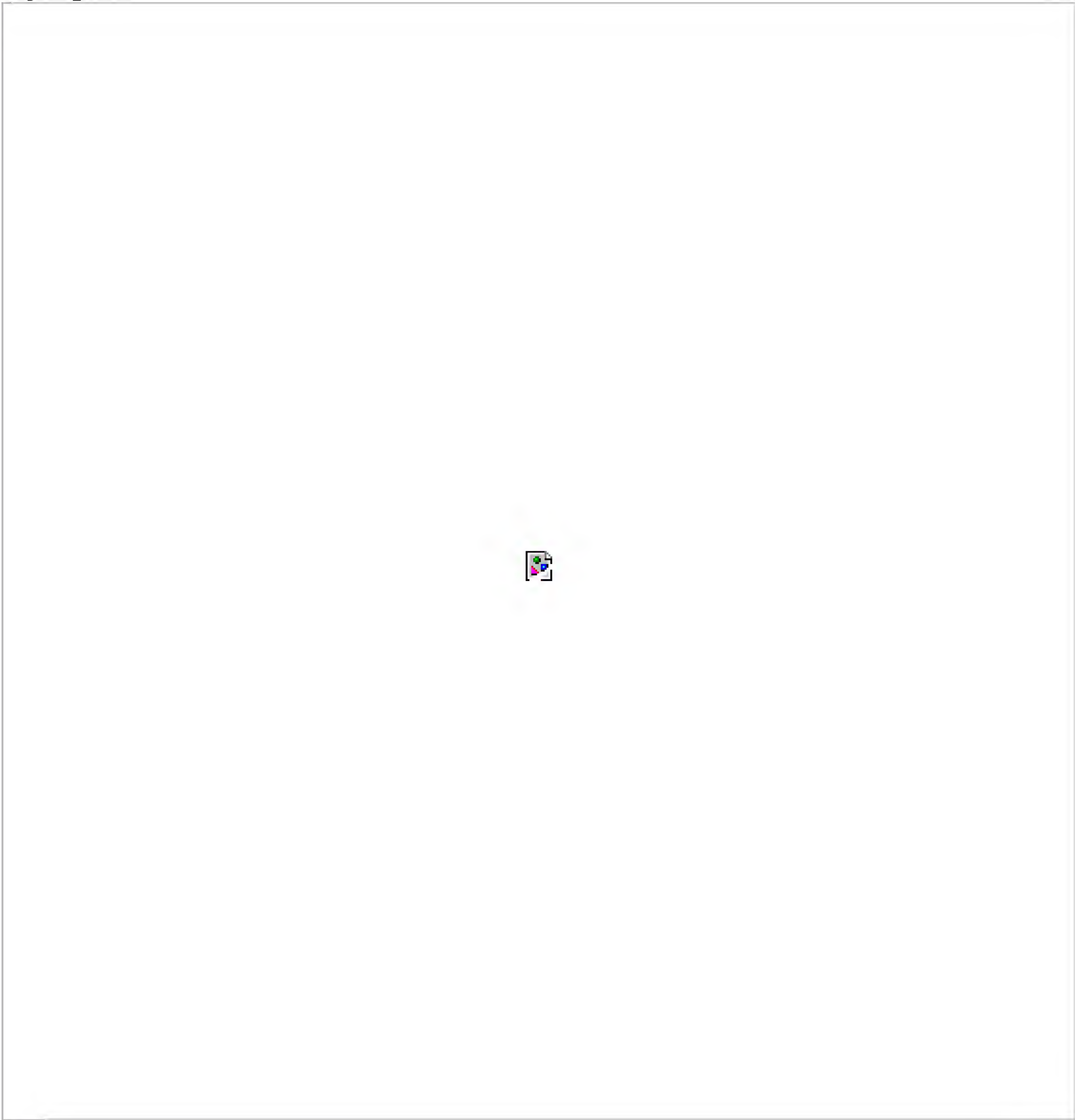
1.-No hay ciclos

2.-De padre a hijos son validas las relaciones de uno a uno a uno a muchos.

El esquema de una base de datos jerárquica se representa como una colección de diagramas de estructuras de árbol. Para

cada diagrama existe una única instancia del árbol de base de datos. La raíz de este árbol es un nodo ficticio. Los hijos de ese nodo son instancias del tipo de registros adecuado.

Ejemplo:



CASOS PARTICULARES







6.3 REGISTROS VIRTUALES.

Dado que en las relaciones muchos a muchos existe demasiada repetición de datos, se maneja el concepto de *registro virtual*. Un registro virtual es aquel que no se escribe físicamente en el medio, sino que es una referencia (liga) a un registro existente en forma previa su representación es :



La razón de utilizar registros virtuales es evitar la repetición de los datos, lo cual traería consigo los siguientes inconvenientes:

-Redundancia, en consecuencia desperdicio de espacio de almacenamiento.
-Alto riesgo de inconsistencia durante las actualizaciones